Towards controllable morphogenesis in large robot swarms

Daniel Carrillo-Zapata[1,2,3], James Sharpe[4], Alan F. T. Winfield[2,3], Luca Giuggioli[1] and Sabine Hauert[1,3]

***Abstract*—Morphogenetic engineering aims to achieve functional, self-organized but controllable structures in human-designed systems. Controlling the structures is crucial if they are to be used for real-world applications. Building on previous work on morphogenesis, in this paper we present a new algorithm, with controllability at its core, for large swarms of simple robots where morphogenesis occurs without self-localization, predefined map, or preprogrammed robots. Controllability is achieved through three parameters that influence the morphogenesis process and create a rich morphospace of quantitatively different shapes. The algorithm was tested in over 2000 simulations and 3 times on real swarms of 300 kilobots. Swarms were able to grow shapes using only local communication, and regrow missing parts when manually damaged. Extra simulations also demonstrated swarms adapting to an obstacle in the environment by getting around it. Results were compared with our previous work on morphogenesis to show how controllability allowed richer shapes. This work represents a step into designing a controllable morphogenesis algorithm towards more functional swarms for real-world applications.**

***Index Terms*—Cooperating Robots, Distributed Robot Systems, Swarms, Morphogenetic engineering, Kilobots**

I. INTRODUCTION

MORPHOGENETIC engineering [1] seeks to form emergent, functional structures using multiple agents through a decentralized, self-organized but controlled process. One of the approaches is to bring the morphogenesis capabilities of biological systems into human-made systems. Inspiration may come from embryo development [2], ants building bridges to cross gaps [3], termites constructing nests for protection from predators [4], or slime mold such as *Physarum polycephalum*, able to modify its shape to forage [5]. From the perspective of swarm robotics, where a collective behavior emerges from a large number of robots following simple rules and interacting locally in a distributed and decentralized manner [6], functional shapes could lead to self-constructing structures able to physically adapt to spatial conditions [7], reconfigurable modular robots able to modify their shape dynamically depending on the task (e.g., area coverage to clean gutters [8]), or patterning at the micro/nano scale for biomedical applications [9]. If morphogenetic swarms are to be deployed in the real world, controllability should be at the core of the design to make them usable.

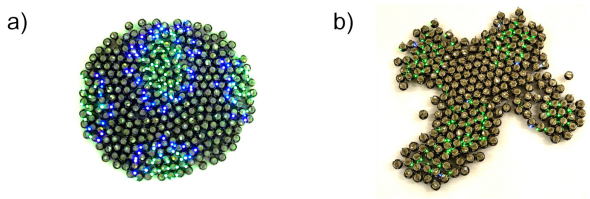

Fig. 1: *a)* Initial configuration of a swarm of kilobots before morphogenesis. *b)* A kilobot swarm during morphogenesis.

Previous work by Rubenstein et al. [10] and Gauci et al. [11] used a top-down mechanism to form shapes with 1024 robots. In their work, although the algorithm was fully decentralized, robots were given a map of the shape, and four preprogrammed robots were needed to dynamically build a coordinate system. Instead, our previous work [12] uses a bottom-up approach to morphogenesis where shapes emerge in a fully self-organized manner from the spontaneous, self-organized, symmetry-breaking phenomena able to produce spatial patterns observed in some biological systems during embryogenesis [13], [14]. The algorithm was demonstrated with swarms of 300 kilobots, showing that swarms were adaptable and resilient to damage. The robots self-organized to create shapes by only relying on local interactions with neighbors and without using any map, coordinate system or preprogrammed seed robots. However, shapes lacked the type of controllability pursued in morphogenetic engineering.

Building on the previous work, in this paper we propose an entirely new, bottom-up morphogenesis algorithm to allow such controllability. The main contribution is the local gradients algorithm for the morphogenesis phase, whereby the swarms self-organize into different shapes using only local interactions and without the need for any map, coordinate system, central control or seed robots. For this approach to be useful in real-world applications in the future, e.g., collective area coverage (a common application in swarm robotics), we should be able to control where the growth is happening, and/or how much shapes are able to grow. This in fact would allow addition of task-specific knowledge into the swarms for human-swarm interaction [15]. As opposed

Manuscript received: February, 24, 2019; Revised April, 19, 2019; Accepted June, 18, 2019.

This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE).

[1]Daniel Carrillo-Zapata, Luca Giuggioli and Sabine Hauert are with the University of Bristol, Bristol, UK daniel.carrillozapata@bristol.ac.uk

[2]Daniel Carrillo-Zapata and Alan F. T. Winfield are with the University of the West of England, Bristol, UK

[3]Daniel Carrillo-Zapata, Alan F. T. Winfield and Sabine Hauert are with the Bristol Robotics Laboratory, Bristol, UK

[4]James Sharpe is with the European Molecular Biology Laboratory, Barcelona, Spain

Digital Object Identifier (DOI): see top of this page.

to [12], here we design a controllable morphogenesis algorithm with entirely new rules, and introduce three parameters that directly impact shape growth. Results were compared with our previous morphogenesis approach to show the richer range of shapes that the algorithm presented here is able to produce.

Collective area coverage has been studied in the areas of mobile sensor networks [16] and robotics [1], [17], [18]. Most work has focused on using multiple robots for area coverage (robots deployed in an environment to maximize area covered for monitoring purposes, for example) or swarm-guided navigation (robots organizing spatially to guide others). However, most assume precise motion and sensing capabilities such as measuring angles to neighbors or localization [19], [20]. Furthermore, simulation or a relatively small number of robots (no more than a few dozens) have been used, and few of them have tested resilience to damage with real robots. Therefore, it is not clear whether they would be applicable for large swarms of simple, noisy robots. In this paper we are only interested in demonstrating the ability of large swarms of minimal robots to grow a wide range of shapes, in a controlled manner, that are scalable to different swarm sizes and resilient to damage, rather than in optimizing area coverage for a particular scenario.

II. METHODOLOGY

We use kilobots [21] to demonstrate our approach. These robots are equipped with two vibrating motors on each side, infrared communication up to 10cm, an ambient sensor and an LED. Even though kilobots lack precise motion and sensing due to their low-cost design, they are an ideal platform to demonstrate collective behaviors.

A general overview of the algorithm is given in fig. 2a. The algorithm is composed of two main phases: Patterning and morphogenesis. During patterning, clusters of attracting robots emerge from random initial conditions by a mechanism of reaction and diffusion of virtual molecules. Patterning is then combined with a guided process of morphogenesis based on local gradients. The emergent clusters are elongated by other robots stopping around them. This provides the swarm with the ability to grow a shape in a structured but completely self-organized fashion. A detailed description of each phase is given below, and a link to the source code is provided at the end of this paper.

A. Patterning

Reaction-diffusion is a mathematical model, which describes the process whereby certain molecules (named *morphogens*) chemically react with each other and diffuse through the space. Under the right conditions, they self-organize into structured patterns from pure random initial concentrations. For this work, a linear approximation of a reaction-diffusion system is used with the same parameters and structure described in [12]. Each robot is programmed with an internal representation of the concentration of two types of molecules—one being an activator (U) and the other an inhibitor (V)—, present in the area defined by the robot. At every timestep, the molecules inside the robots react with each other, increasing or decreasing their concentration u and v. Diffusion is approximated through message passing by transferring and receiving molecules from their neighbors. The change in their concentration is governed by the following linear equations and conditions encoded in the robots:

$$\frac{\Delta u}{\Delta t} = Rf(u,v) + D_u\nabla^2 u$$
$$\frac{\Delta v}{\Delta t} = Rg(u,v) + D_v\nabla^2 v$$
$$f(u,v) = (Au + Bv + C) - \gamma_u u$$
$$g(u,v) = (Eu - F) - \gamma_v v$$
$$0 \leq (Au + Bv + C) \leq synU_{max}$$
$$0 \leq (Eu - F) \leq synV_{max}$$

Coefficients are: $R = 160$, $D_u = 0.5$, $D_v = 10$, $A = 0.08$, $B = -0.08$, $C = 0.03$, $E = 0.1$, $F = 0.12$, $\gamma_u = 0.03$, $\gamma_v = 0.06$, $synU_{max} = 0.23$, $synV_{max} = 0.5$, $\Delta t = 5{\times}10^{-5}$.

After approximately 10 minutes, a stable pattern emerges in the robots from completely random initial concentrations. Robots are able to automatically detect the pattern is stable by adding up the change in their own molecules concentration during windows of two minutes. If the accumulated change is below a threshold (1.5 units), they transition. The choice of this threshold was based on experimentation in simulation on the accumulated change when the pattern was stable versus when it was in development during the early stages. In this self-organized phase transition, the pattern can be seen as a form of automatic task allocation. If the concentration of the activator molecule inside a robot is above a certain threshold (3 units in this paper), the robot is considered to be an attractor in the next phase. As a result of patterning, clusters of attracting robots appear on the edge of the swarm due to the specific set of parameters of the reaction-diffusion system implemented on the robots (see Supplementary Materials of [12] for more details about the reaction-diffusion system implemented here). Robots with a lower concentration than the threshold are considered non attractors. Hence, clusters are made of attracting robots only.

It is worth mentioning that another patterning strategy could potentially be used (for example, small clusters made of source robots spread out by a minimum distance from each other). Here we chose reaction-diffusion patterning due to its self-organized nature, and its ability to create spots on the edge of the swarm with the coefficients given above. In the unlikely event of no spots on the edge, robots could rerun patterning as many times as desired (not implemented here).

B. Morphogenesis

In this phase, reaction-diffusion stops and robot movement starts. For a shape formation behavior to occur in the robots, they have to move towards other areas in the environment while maintaining connectivity at all times. Branching-out behavior happens with the help of robots being attracted to the clusters of attracting robots that have arisen from

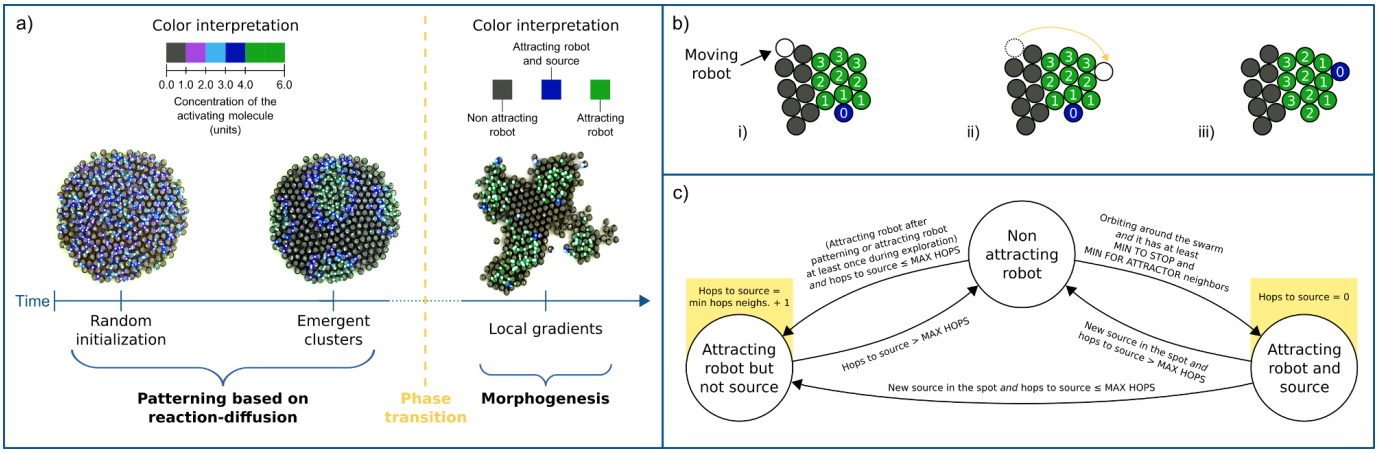

Fig. 2: Overview of the algorithm. *a)* Two-phase schematic where morphogenesis occurs after patterning. Color interpretation is different for the two phases. *b)* Part of a swarm with a cluster of attracting robots (in dark blue and green) and non attracting robots (in white and gray). A moving robot (*i*)) stops next to the cluster (*ii*)) and replaces the source of the signal (*iii*)). Hops are then readjusted to the new source. The hop-based threshold for belonging to the cluster results in the cluster remaining at the tip. *c)* Finite state machine for local gradients.

patterning. Those clusters act as virtual docking points for other robots. When a moving robot senses enough adjacent attracting neighbors, it stops and may become part of the attracting cluster to continue attracting other robots (more details given in next subsection). Connectivity in the swarm is guaranteed by edge-following movement, originally described in [10] and replicated in [12]. Edge following allows the robots to move around the swarm while maintaining a fixed distance to the nearest neighbor (in our case, 45mm). Moreover, only the robots on the edge of the swarm which do not belong to any cluster are allowed to move. Edge detection is based on a cohesion metric obtained by comparing the number of neighbors of the robot with the average number of neighbors of its neighbors. Using information from neighbors produces a more robust measure of cohesion [22]. Direction of movement, i.e., clockwise or anti-clockwise, is randomly chosen with uniform probability $p = 0.5$ when the robot is initialized, and is fixed throughout the entire algorithm.

Growth is achieved by having the attracting clusters at the tip of the branch at all times, as well as maintaining the size of the cluster within certain limits. If the clusters were too large or grew in size, other robots would be attracted to the base of the branch, hence, not producing the desired elongating behavior. Robots fulfil those two conditions by creating a local gradient in the cluster. The last robot which arrives to the cluster (at the edge of the swarm) becomes the source of a signal which is transmitted inside the cluster of attracting robots to establish a hop-based gradient from itself. Robots calculate the minimum number of communication hops to reach the source of the gradient either directly or indirectly through their neighbors. A neighbors' table is used to store the information from a maximum of 20 neighbors. The information stored is the random, unique, local ID from each neighbor, its distance, state, number of neighbors, whether it is an attracting robot, whether it is the source of a signal, the ID of the source of the signal in its cluster (if in any), the minimum number of hops to the source of the signal (infinite if not in a cluster), a timestamp of when the last message from this neighbor was received (entries older than two seconds are deleted), and its molecules concentration (for the patterning phase). Each robot has a 2-byte random, unique, local ID to make the algorithm completely scalable with respect to number of robots (see Supplementary Materials of [10]). If two neighbors share the same ID by chance at some point, they will generate random IDs until none of them share the same one—therefore, unique in their neighborhood.

By combining a maximum number of hops to the source of the gradient with a mechanism whereby the last robot arriving to the cluster becomes the only source of the gradient signal, the robots self-organize to maintain the cluster at the tip of the branch (see fig. 2b). These clusters are indeed "nearly circular regions of controlled size", as pointed out by Mamei et al. [23]. The robots with a number of hops to the nearest source higher than the threshold do not continue with the hop count, i.e., the signal is only transmitted inside the cluster to create a local gradient. Since such signal is not transmitted to the whole swarm, the process is completely scalable with respect to the number of robots in the swarm. Depending on the dynamics, any robot could potentially carry out any of the roles. Furthermore, no map, coordinate system or seed robots are required.

C. Controllable morphogenesis

The goal in this paper is to control the morphogenesis process while allowing self-organization. This is achieved by having a range of variables that control the shape by modifying the local gradients. Six variables were initially proposed. A regression model based on random forests [24] was used to select the most relevant variables. The performance measure was the total area covered during 20 repetitions of each combination of variables with different random seeds, as

explained in section III. The following three variables were identified by the algorithm as the best for controllability:

- *Maximum number of hops to the source of the local gradient (max hops)* It defines a hop-based threshold to belong to a cluster of attracting robots. This variable is directly linked to the maximum size of the cluster.
- *Minimum number of attracting neighbors to stop (min to stop).* It defines the minimum number of neighbors inside an attracting cluster which a moving robot must sense to stop next to the cluster. This variable affects the location of the cluster where the robot stops, e.g., more on the sides or in the middle.
- *Minimum number of attracting neighbors to become an attractor (min for attractor).* It is the minimum number of attracting neighbors which a robot (either in the patterning or morphogenesis phase) must sense to become part of the attracting cluster. This variable controls the minimum size of a cluster.

Fig. 2c shows a diagram with the state machine for local gradients using the variables above. It is worth mentioning that there are other parameters and transitions not shown in the state machine for simplification. For instance, other robots on the edge of an attracting cluster might become sources with a certain uniform probability in case the source of the signal disappears. In addition, each robot has a counter to avoid leaving a cluster straight away in case the minimum number of hops to the closest source exceeds the *max hops* threshold. These conditions add robustness to a system of unreliable, noisy robots. Full details can be seen in the source code and Materials and Methods of [12].

It is also worth highlighting that the three variables are independent of each other, in particular *min to stop* and *min for attractor*. A robot has only to satisfy *min to stop* for it to stop. When it satisfies *min for attractor* it joins the cluster to be the source of the signal after it has stopped. In case there are not enough attracting neighbors after stopping, the robot would still be non attracting. Therefore, there would not be a transition in the state machine shown in fig. 2c.

III. RESULTS

While a full analysis of reaction-diffusion patterning, adaptability to growth and self-healing is explored in [12], here we focus on the controllability and performance aspects of the morphogenesis algorithm here proposed.

A. Morphogenesis in simulated swarms

A parameter sweep across all combinations of plausible values was performed using simulator Kilombo to understand the effect of the morphogenesis variables quantitatively. Swarms of 250 simulated kilobots were placed together in circular initial shape at the centre of a squared window of 850x850 pixels, initially covering about 8% of the total area. The choice of this number of simulated agents was made as a trade-off between the number of repetitions of each combination and running time of each simulation. Performance of the different combinations was measured by calculating the total area of the window which the robots covered in an interval equivalent to 8 hours in real time (the amount of new area covered was continually stored at every timestep of the simulations). The purpose of the metric here was to measure how shape growth could be controlled in terms of by how much they could grow driven by the different combinations of variables. This controllability will be necessary to ultimately use morphogenetic engineering for specific applications in the future.

TABLE I: Tested values of the variables for controlling morphogenesis and the best values found.

Variable name	Values tested	Best values
Max hops	0, 1, 2, 3, 4	0, 1
Min to stop	1, 3, 5, 7, 9	5, 7, 9
Min for attractor	1, 2, 3, 4	1, 2, 3

To decrease the reality gap between simulated and real kilobots experiments, noise was added to both communication and motion. Each message had a 0.7 probability of being received correctly and not being discarded. Communication noise directly proportional to the distance between two robots was added with a Gaussian distribution of mean zero and standard deviation defined by: $\sigma_{distance} = \frac{d-min_d}{max_d-min_d}$. In the previous equation, d is the distance measured by the robot, whose minimum is $min_d = 34$mm (the diameter of the kilobots) and maximum is $max_d = 85$mm (the maximum communication range allowed). As far as motion is concerned, every robot was initialized with a fixed bias in its angular velocity randomly chosen from a Gaussian distribution of mean zero and standard deviation $\sigma_{\omega bias} = 0.01$. Moreover, another source of noise was added in the form of a Gaussian distribution of mean zero and standard deviation $\sigma_{\omega noise} = 0.005$ every time the simulator updated the movement of the robots. Noise values were chosen based on experimentation with real kilobots. Robots which separated from the swarm due to noise were not taken into account for area coverage. Robots must have at least four neighbors to take part in the metric. The rationale behind is that lost robots would be unsuccessful in maintaining communication with the rest of the swarm, therefore, not contributing to a common shape.

Table I shows the different values tested. A preliminary exploration was performed to identify the meaningful ranges for the variables. From five *max hops* upwards, swarms reached a static configuration due to most of the robots on the edge becoming part of a cluster soon after the start of morphogenesis. Regarding *min to stop*, robots on the edge usually sense an average of nine neighbors because density decreases compared to being in the middle of the swarm. Finally, the parameters of the reaction-diffusion system sometimes generate clusters of four attracting robots after patterning. If *min for attractor* was higher, those clusters could not be formed, hence, losing attracting robots and chances for substantial growth.

For every combination, 20 repetitions with different seeds were performed. The same set of unique seeds was used for all combinations. Therefore, patterns emerged from reaction-diffusion were the same for all combinations (but not repetitions). This allowed to properly study the effect of the variables

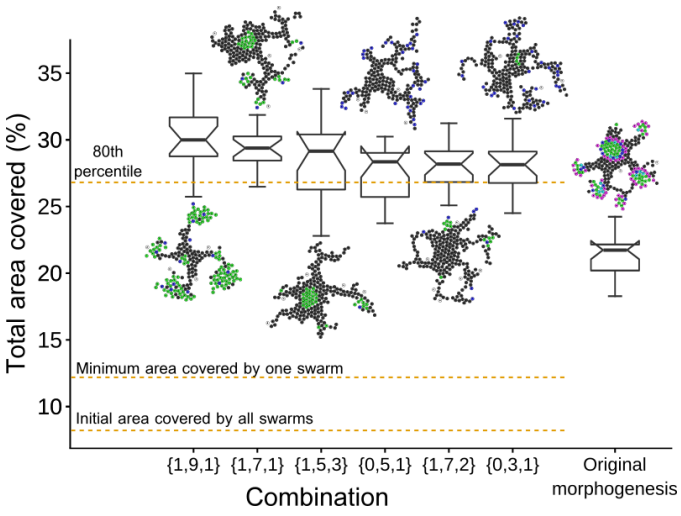

Fig. 3: Box plot showing the best combinations of variables and the performance of the original morphogenesis algorithm [12]. The final shape of the best run for every combination is shown as well. Combinations are sets of {*max hops, min to stop, min for attractor*}. Dotted lines represent the corresponding area covered as described above them.

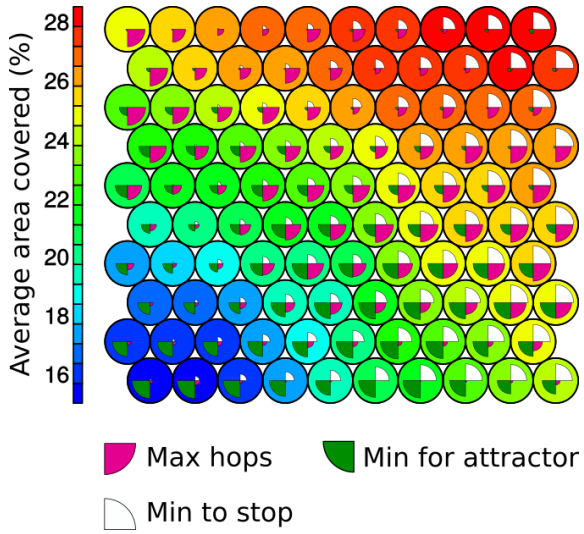

Fig. 4: Self-organizing map with all combinations of the variables and a heat map of average performance across repetitions. Bigger wedges correspond to bigger values.

on morphogenesis under the same conditions. A total of 2000 simulations was performed.

The 95% confidence interval for the median of the total area covered by the simulated swarms was calculated for each combination. The combinations with the highest, overlapping intervals of the median were identified (fig. 3). As a result, 6 out of 100 combinations were found to produce the highest performance in terms of total area covered. A one-way analysis of variance (ANOVA) was done over the six combinations to test whether they were statistically different. This test rejected the null hypothesis ($F = 4.6281$, p-value < 0.0007), meaning that there were some differences in the means between those six combinations (with a 95% confidence level). A post-hoc Tukey test showed that the following combinations were statistically different at a 95% confidence level: {1,9,1} and {0,3,1} (p-value < 0.015), {1,9,1} and {1,7,2} (p-value < 0.023), {1,9,1} and {0,5,1} (p-value < 0.0016), {1,7,1} and {0,5,1} (p-value < 0.045).

Substantial growth did occur in simulated swarms. In the six best combinations of morphogenesis variables, swarms covered between three and four times their size while maintaining connectivity at all times, even if an average of 10% of robots got lost or became detached from the swarm by the end of the simulations. An extra 20 repetitions with the same set of seeds used for each combination were done to compare the morphogenesis algorithm presented here with the one described in [12]. The area covered by these swarms is also shown in fig. 3.

Fig. 3 also demonstrates that morphogenesis variables have a different effect depending on how they are combined. Even though there is no statistical difference among means of some of the best combinations, a difference between them and the rest can be observed. Support comes from the fact that more than 50% of the repetitions (the median) of each combination shown in the figure covered a bigger area than 80% of all combinations and repetitions (the 80th percentile). A self-organizing map with all combinations was produced to further explore the effect of such combination of variables (fig. 4). The fact that the combinations space is relatively small (100 combinations) allows to show a combination per node. For each one, its weighted vector of the three morphogenesis variables is shown, as well as the corresponding color in the heat map of average area covered across all repetitions. As a conclusion, low *max hops*, medium/high *min to stop* and low/medium *min for attractor* maximized area covered by the swarm over time.

Scalability of the algorithm was also studied in simulation. Combination {1, 5, 3} was repeated 20 times with 100 and 1000 robots, and different random seeds, during a length of time equivalent to 8 hours in real time. Those simulations were compared with the results from the same combination using 250 robots. Results are shown in fig. 6. The curves present a similar behavior over time, meaning that the inclusion of more robots does not negatively affect the performance of the algorithm, hence it is scalable.

Finally, two extra simulations with swarms initialized with different random seeds in a scenario with the same obstacle below them were performed. The obstacle was thick enough to avoid robots communicating through it. Results can be seen in fig. 5. In both cases, robots moved around the obstacle by surrounding it and reconnecting with the clusters in the growing branches on the right-hand side of the obstacle. This demonstrates that swarms were able to adapt and continue exploring in the presence of an obstacle.

B. Morphogenesis in real swarms

Three repetitions using combination {1, 5, 3} were performed using real swarms of 300 kilobots to test whether morphogenesis occurred in real swarms (see supplementary video). This combination was chosen arbitrarily from the three

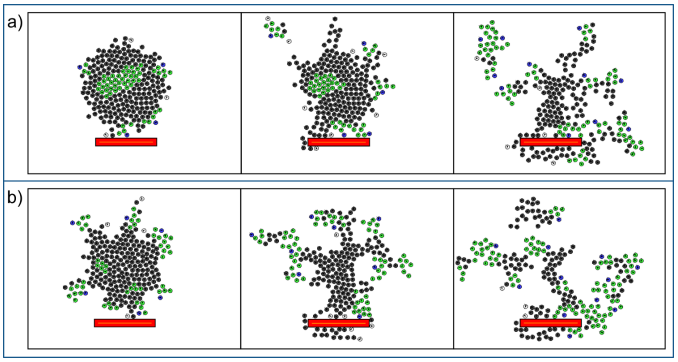

Fig. 5: Response to obstacles. Two different swarms (*a* and *b*) get around an obstacle situated below them.

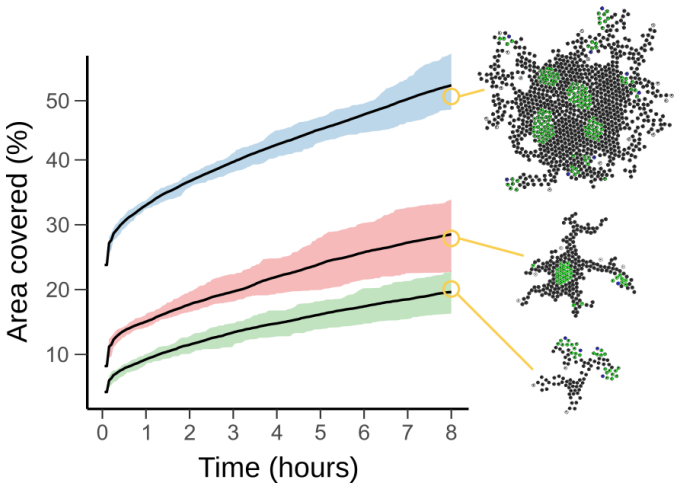

Fig. 6: Scalability of the morphogenesis algorithm in simulation with combination {1, 5, 3} repeated 20 times. The average, minimum and maximum area covered is shown with 100 robots (bottom, green ribbon), 250 robots (middle, red ribbon) and 1000 robots (top, blue ribbon). One final shape is shown next to the corresponding curve.

best combinations with no proved statistical difference, and because their growth curves were very similar. Robots were arranged in the same conditions as in simulation, i.e., in a circular initial shape at the centre of a 2*x*2m arena. Initially, the swarms covered 6% of the whole area, approximately. Experiments were run for 3 hours and a half given battery autonomy. The patterning phase took between 10 and 15 minutes. Results of the area covered by the swarms over time are shown in fig. 7. A similar performance can be seen across the three runs. Area covered increased to as much as twice the size of the swarms by the end of experiments. This demonstrates that the algorithm proposed in this paper also grows shapes in real robots.

Two additional experiments were performed to demonstrate the resilience of the algorithm to damage. For these experiments, the self-organized phase transition was improved (window length was reduced to one minute, and robots could also transit to the morphogenesis phase in case they had at least two neighbors already in that phase). In the first experiment, part of a branch was removed after one hour from the beginning of the experiment, leaving only a few robots in the cluster of that branch. In the second experiment, a whole branch and cluster were removed after one hour and a half. The robots manually taken from the swarm were completely removed from both experiments. As seen in fig. 8a, the branch regrew completely and extended its length by the end of the experiment (about 3 hours and a half in total). In the second experiment shown in fig. 8b, the branch did not grow back. Instead, branches on both sides of the cut could develop further after the same amount of time as in the first experiment. The result of these experiments show that the proposed morphogenesis algorithm is able to recover from damage, either by regrowing broken branches or reallocating more robots to the vicinity.

IV. DISCUSSION

Even though the area covered is similar across the six best variable combinations in simulation, shapes are qualitatively different. Each morphogenesis variable has a different impact. However, it is the combined effect of the three of them that produces different shapes. To understand such effect, simulations were done using the extreme values for each variable (for *max hops*, a value of one instead of zero was used). Same seed was used across simulations. An analysis of the swarm morphologies was done using two different morphometrics to measure the effect of the variables quantitatively. Morphometrics were shape index, a metric describing how much a shape deviates from a circle, and the minimum number of characterizing points, which measures the simplicity of a shape (see Supplementary Materials of [12] for more details). Results of the morphospace created by the shapes produced by the morphogenesis algorithm presented in this paper as well as four shapes produced by the original morphogenesis algorithm presented in [12] can be seen in fig. 9. Indeed, having morphogenesis variables controlling the morphogenesis process produces greater richness of shapes compared to our previously published morphogenesis algorithm. This could be advantageous in situations where the swarm has to adapt to the environment, e.g., when performing collective area coverage. By modifying the morphogenesis variables, the swarm could produce a whole range of different growths depending if wider/thicker or further/thinner reach is required for the task. Another advantage of the algorithm presented in this paper is that it could be easily extended to allow robots to create new clusters of attracting robots during morphogenesis where needed, e.g., to grow new branches in case the swarm is completely blocked by an obstacle.

With respect to *max hops*, a higher number does not produce branching out as effectively. The reason is the size of the clusters. If the hops threshold is high, one source can include many robots in the cluster. The edge then gets saturated with robots in a cluster, causing a wider expansion until the point where no robots on the edge move because they belong to a cluster. As a consequence, the lower *max hops*, the better for branching out. In the case when *max hops* is zero, sources of the signal are the only ones in their own cluster. Therefore,

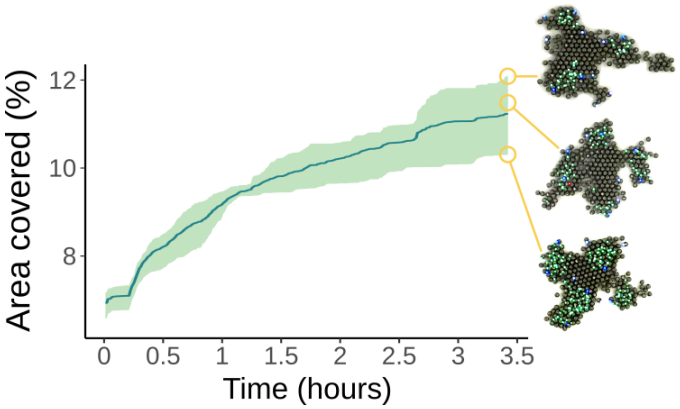

Fig. 7: Percentage of area covered by swarms of 300 kilobots. Blue line is the average of three experiments with the same parameters. Minimum and maximum area covered by these swarms is shown as a green ribbon.

moving robots will stop near sources, depending on the other values. This causes a branching out effect as well, as seen in fig. 3. Another crucial aspect regarding *max hops* is that only the robots inside a cluster are allowed to belong to it. If instead all robots meeting the *max hops* criteria were allowed, this would have a similar effect to the behavior seen when *max hops* is high. Eventually, more and more robots would become part of a cluster, hence blocking the robots on the edge of the swarm from moving.

A low value of *max hops* alone does imply branching out. Due to communication noise, for example, a robot might stop next to a cluster independently of *min to stop*. If *min for attractor* is low enough, the robot will become part of the cluster. The lower this threshold, the more likely it is that stopping robots will add to clusters, thus, helping morphogenesis. If the value is high (or *min for attractor* $>$ *min to stop*), robots might stop but not join the cluster because there would not be enough attracting neighbors. This means that less branch growth would be achieved, hence, stopping morphogenesis. As a consequence, the lower *min for attractor*, the better for branching out.

When *max hops* and *min for attractor* are low but *min to stop* is high, branches are thinner and longer. This has to do with the tendency of this combination to create bifurcations. With these conditions, sources tend to appear on the sides of the cluster when it reaches a certain size. A high number of sources can also be seen when the three variables are low. However, the behavior is slightly different. In this case, robots in the middle of a cluster may eventually leave it after a bifurcation. But as soon as those robots move again, they will stop because of their attracting neighbors and the low threshold for the stopping condition. Therefore, lost parts of the cluster tend to be recovered straight away, causing a wider growth. When *min to stop* is high, the robots in the middle leaving the cluster will orbit a longer distance around it, helping bifurcation. As a consequence, the higher *min to stop*, the better for branching out. The three conclusions discussed in this subsection match the results of the six best combinations shown in fig. 3.

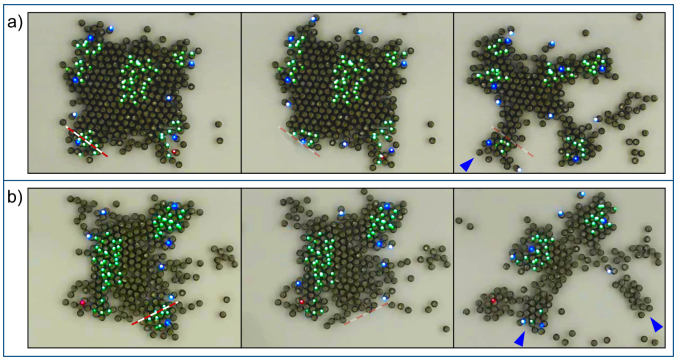

Fig. 8: Resilience to damage. *a)* Half a cluster is removed (dashed line), and regrows. *b)* One entire cluster is removed (dashed line), and branches grow on both sides.

In the real swarms, growth was produced in a similar fashion across the three runs as a result of the morphogenesis variables. After patterning created four clusters on the edge of the swarm, four branches of similar thickness grew. Results show that the area covered increased over time repeatedly in the three runs. This confirms the morphogenesis process can also be controlled in large swarms of simple, real robots. However, swarms covered less area in the same amount of time compared to simulation, as seen in the middle curve from fig. 6. This shows that there is still a reality gap between simulation and the real kilobots. Due to their very noisy behavior (especially when it comes to motion), it is hard to model them accurately in simulation. The addition of noise helps crossing such gap [25], but it does not manage to represent reality completely. A more accurate noise model will be considered in future work.

Finally, swarms have been demonstrated to be able to recover from damage. In the case of part of a cluster being removed (fig. 8a), the branch could regrow because the damaged cluster could still attract other robots that eventually became part of the cluster and restored it. On the contrary, if the whole cluster disappeared (fig. 8b), it could not grow back. Instead, such area in the swarm was freed up so that robots on the edge could travel to nearby clusters in both directions. This produced more growth in other branches. A mechanism to run patterning again when the swarm loses a certain number of clusters could help recover from more substantial damage, such as losing all branches. This will be studied in future work, as well as a mechanism to prevent the swarm from splitting.

V. CONCLUSION

A new morphogenesis algorithm based on local gradients has been designed for large swarms of simple, noisy robots. Robots self-organize to grow controllable shapes while maintaining the communication network without the need for a map of the shape, a coordinate system or preprogrammed seed robots. Morphogenesis variables to control the shapes have been identified by performing over 2000 simulations. Results have demonstrated the rich morphospace that these variables are able to produce. This has the potential to enable adaptive growth dynamically if required.

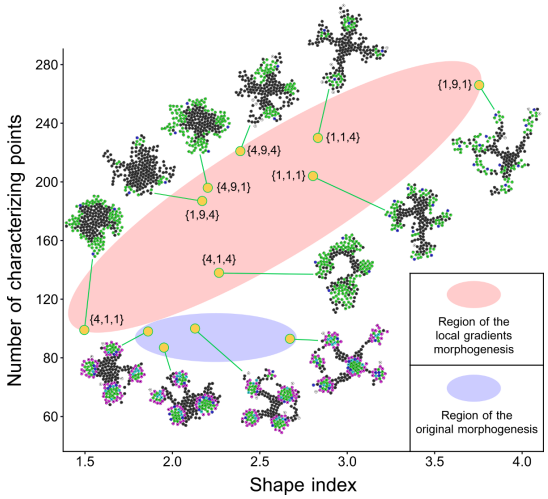

Fig. 9: Morphospace produced by different combinations of variables (labelled as {*max hops, min to stop, min for attractor*}) and the original morphogenesis algorithm [12].

One of the best combinations found was tested three times on real swarms of 300 kilobots. This combination produced similar results repeatedly across the three runs. Furthermore, two experiments with real swarms of 300 kilobots were conducted to demonstrate that the algorithm is resilient to damage. In one experiment, part of a branch was cut off after one hour of morphogenesis, and managed to grow back completely. In another experiment, a complete branch was removed, resulting in longer growth of the branches on both sides of the missing branch. Extra simulations with different number of robots (100, 250 and 1000) have shown that the algorithm is scalable, thanks to being only based on local information. Swarms grew a shape in a similar manner independently of the number of robots. Finally, another two simulations have shown how the swarms can continue growing in the presence of an obstacle.

This work represents an example of bottom-up, self-organized, controllable, scalable, adaptable and resilient morphogenesis algorithm tested on large swarms of real robots, getting a step closer to the goals of morphogenetic engineering. Our future aim is to extend and make this approach functional for suitable swarms released in disaster scenarios [26], [27] to explore the environment and deploy exit routes by creating a communication/visual chain to guide other robots or humans (swarm-guided navigation [17]), for example. Indeed, we are currently running use case studies with firefighters to understand where this approach could be useful for them. Further work will focus on shortest-path creation to connect objects of potential interest.

SOURCE CODE

https://github.com/Danixk/Functional_morphogenesis

ACKNOWLEDGMENT

D. Carrillo-Zapata thanks S. Jones for the noise model in simulated kilobots, and H. Chandler and R. Garcia-Martinez for their help during experiments.

REFERENCES

[1] R. Doursat, H. Sayama, and O. Michel. *Morphogenetic engineering: toward programmable complex systems*. Springer, 2012.
[2] J. D. Murray. *Mathematical Biology: I. An Introduction*. Interdisciplinary Applied Mathematics. Springer, 2011.
[3] C. Anderson, G. Theraulaz, and J-L. Deneubourg. Self-assemblages in insect societies. *Insectes sociaux*, 49(2):99–110, 2002.
[4] C. Noirot and J. P. E. C. Darlington. *Termite Nests: Architecture, Regulation and Defence*, pages 121–139. Springer, 2000.
[5] C. R. Alvarado and S. L. Stephenson. *Myxomycetes: Biology, Systematics, Biogeography and Ecology*. Academic Press, 2017.
[6] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In Erol Şahin and William M. Spears, editors, *Swarm Robotics*, pages 10–20. Springer, 2005.
[7] R. Doursat, H. Sayama, and O. Michel. A review of morphogenetic engineering. *Natural Computing*, 12(4):517–535, 2013.
[8] M. Yim, W-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.
[9] S. Hauert and S. N. Bhatia. Mechanisms of cooperation in cancer nanomedicine: towards systems nanotechnology. *Trends in Biotechnology*, 32(9):448–455, 2014.
[10] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.
[11] M. Gauci, R. Nagpal, and M. Rubenstein. *Programmable Self-disassembly for Shape Formation in Large-Scale Robot Collectives*, pages 573–586. Springer, 2018.
[12] I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert, and J. Sharpe. Morphogenesis in robot swarms. *Science Robotics*, 3(25), 2018.
[13] L. Marcon and J. Sharpe. Turing patterns in development: what about the horse part? *Current Opinion in Genetics and Development*, 22(6):578–584, 2012.
[14] A. D. Economou, A. Ohazama, T. Porntaveetus, P. T. Sharpe, S. Kondo, M. A. Basson, A. Gritli-Linde, M. T. Cobourne, and J. B. A. Green. Periodic stripe formation by a turing mechanism operating at growth zones in the mammalian palate. *Nature genetics*, 44(3):348, 2012.
[15] A. Kolling, K. Sycara, S. Nunnally, and M. Lewis. Human-swarm interaction: An experimental study of two types of interaction with foraging swarms. *Journal of Human-Robot Interaction*, 2(2):103–129, 2013.
[16] B. Wang, H. B. Lim, and D. Ma. A survey of movement strategies for improving network coverage in wireless sensor networks. *Computer Communications*, 32(13-14):1427–1436, 2009.
[17] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.
[18] H. Oh, A. R. Shirazi, C. Sun, and Y. Jin. Bio-inspired self-organising multi-robot pattern formation: A review. *Robotics and Autonomous Systems*, 91:83–100, 2017.
[19] C. Zhu, L. Shu, T. Hara, L. Wang, S. Nishio, and L. T. Yang. A survey on communication and data management issues in mobile sensor networks. *Wireless Communications and Mobile Computing*, 14(1):19–36, 2014.
[20] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel. Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks*, 58:254–283, 2014.
[21] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298, 2012.
[22] A. F. T. Winfield and J. Nembrini. *Emergent Swarm Morphology Control of Wireless Networked Mobile Robots*, pages 239–271. Springer, 2012.
[23] M. Mamei, M. Vasirani, and F. Zambonelli. Experiments of morphogenesis in swarms of simple mobile robots. *Applied Artificial Intelligence*, 18(9-10):903–919, 2004.
[24] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
[25] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.
[26] S. Hauert, J-C. Zufferey, and D. Floreano. Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32, 2009.
[27] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen. Search and rescue robotics. In *Springer handbook of robotics*, pages 1151–1173. Springer, 2008.